「정보시스템연구」 제9권 제1호
한국정보시스템학회
2000년 6월, pp . 217-234

인터넷 기반 디자인 및 생산지원 분산환경 프로세스관리 기법 연구*

박 화 규**

<목 차>

I. 서 론

단위 시스템이 복잡화 함에 따라 생산성 향상이 주요한 문제로 부각되어가고 있다. 엔지니어링 부문에 있어 CAD/CAM/CAE/CAPP 등의 도구와 전산 시스템의 발달로 제품의 설계, 개발 및 제조에 관한 생산성은 높은 속도로 나아지고 있으나 이에 따라 산출되는 엔지니어링 데이터가 Bottle-Neck으로 작용하여 원격지의 타 부서와 Communication이 이루어지지 못하고 있으며 작업흐름 관리가 원활하게 지원되지 못하고 있다. 이에 따라 프로세스의 개혁 요구가 대두되고 있으며 이를 위한 솔루션으로 Internet Based Process Management 시스템이 등장하게 되었다. 이전의 프로세스 관리는 ERP (Enterprise Resources Planning)와 제품관리 중심이었으나 2000년대에 오면서 프로세스 Cycle Time을 단축시키고자 하는 노력이 계속되고 있다. 이는 생산 현장 중심의 체계에서 제품 설계로 촛점이 옮아가고 있음을 의미하며, 설계의 단축을 위해서는 체계적인 데이터의 관리가 필요하다는 것을 인지하게 된 것이다.

* 본 논문은 Michigan State University와의 국제공동연구로 진행되었음.
** 경동대학교 경영학부 MIS 전공교수

설계 프로세스에 대한 활발한 연구가 현재 진행 중에 있고 일부 상용 소프트웨어가 발표되고 있으나 아직은 많은 연구가 필요한 상태이다. 지금까지 설계 프로세스에 대해 연구되고 있는 내용을 살펴보면 설계 지원 툴로서 Di Janni [Di Janni, 96]는 확장된 Petri Nets를 이용해 고정된 Work Flow를 모델링하는 방법을 제시했고, VOV 시스템 [Casotto Et Al., 99]은 설계자가 툴을 실행할 때 그 Sequence를 기록할 수 있게 했다. 이 시스템은 입력 파일이 수정되면 무효화된 출력 파일 혹은 이전의 툴 실행을 통해 데이터 일관성(Data Consistency)을 유지하게 하였다. 이 외에도 설계 프로세스 중에 실행될 툴을 선정하는 시스템으로 Forward Chaining 방법을 이용해 계획그래프 (Plan Graph)를 산출하는 ADAMS [Knapp Et Al.] 가 있고 설계 프로세스 계획산출을 위해 계층적 전략방법을 이용하는 Minerva [Jacome Et Al.]와 OCT Task Manager [Chiueh Et Al.]가 있다.

또한 설계 프로세스의 지식 처리를 위해 유관 지식들 간의 협조나 상충성을 해결하기 위한 연구로써 Case-Based Reasoning, Agent-Based Approach 및 Blackboard Approach 등의 연구가 활발히 이루어지고 있다. 일본의 경우도 최근 설계에서 AI 기법의 연구와 설계방법론에 대한 연구가 활발해짐에 따라 최근 일본 기계학회에서 설계, System 부문을 설치하여 컴퓨터를 이용한 설계 고도화에 관한 연구를 수행 중에 있다.

이러한 많은 시스템들이 기 개발 혹은 진행 중에 있으나 전반적으로 다음과 같은 문제점을 내포하고 있다. Formalism의 부족경우 Workflow는 많은 예외성이 존재하고 Run Time시 흐름의 재 정의가 불가피하다.

본 연구에서 제안된 Framework은 프로세스의 Formal Representation에 기반을 두었으며 이는 프로세스의 행위를 분석하고 예측을 가능하게 한다. 프로세스 관리 시스템에서는 설계 프로세스를 표현하기 위한 Process Grammar를 개발하고 관련 데이터들 간의 일관성을 유지하면서 설계 프로세스에서 발생되는 Activity들의 관리 통제 기능을 개발하였다. 구체적인 개발 사항은 다음과 같다.

Process Grammar : 설계 프로세스를 정의하기 위한 Grammar를 개발하고 이를 통해 설계의 계층적인 Decomposition 과정 및 설계 데이터와 설계 업무 간의 종속성을 표현하였다.

Process Browser: 설계 프로세스의 관리 및 통제를 위한 Browser가 개발되었다. 이는 Task Library, Production Library, Design Data Library 및 Process Library를 포함하며 각 Library는 Generalization/ Specification 계층 구조로 구성하였다.

Graphical Editor: Process를 편집하기 위한 Graphical Editor를 개발하였다. 이는 Event를 식별하거나 Process를 동적으로 편집하거나 또는 Process를 저장하기 위해 이용될 수 있다.

본 연구에서는 설계 프로세스 관리를 위한 Framework을 개발함으로써 설계를 효율적으로 수행하고 설계의 개발 기간을 대폭적으로 단축할 수 있다. 제안된 Framework은 다음과 같은 장점을 갖고 있다.

Formalism: 설계 프로세스의 관리를 위한 이론적 기반을 구축하여 이를 통해 본 시스템이 다른 방법론과 함께 어떻게 운용될지를 분석할 수 있다.

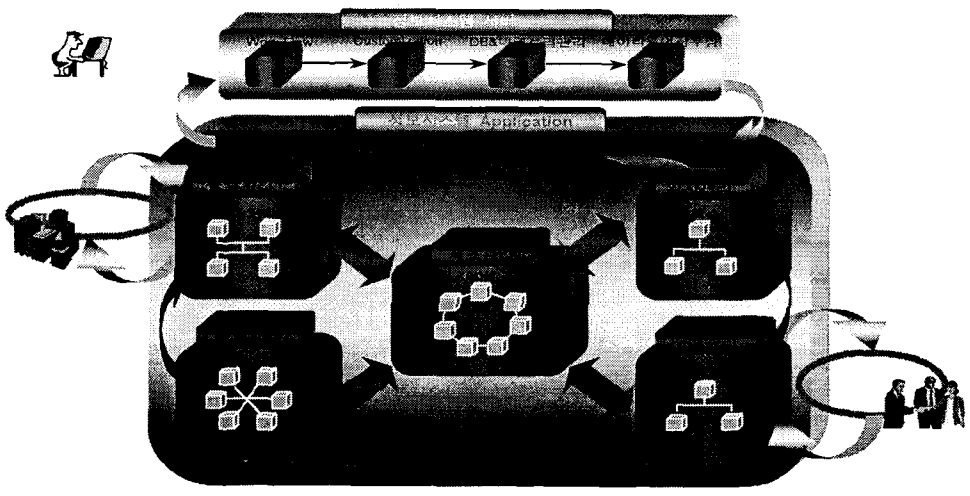
Parallelism: : 몇 개의 대안을 동시적으로 탐색할 수 있게 해 준다. 이는 설계자로 하여금 컴퓨터 자원을 보다 잘 활용할 수 있게 함으로써 설계 시간을 줄일 수 있다.

Reusability: 설계 프로세스가 저장되고 수정되고 재사용될 수 있다. 재 설계 과정과 이용되는 Tool은 이전 Component의 설계 데이터에 크게 의존하며 이 때 제시된 Process Grammar는 이러한 관련성을 얻고 재설계 활동에 대한 지침을 제시하는데 특히 유효하다.

Flexibility: 제시된 Framework은 방법론(Methodology Specification)과 수행환경(Execution Environment)를 명확하게 구분한다.

User Friendliness: Framework은 Internet Web 상에서 GUI환경으로 사용자와 Interactive하게 설계 프로세스를 결정한다.

그림 1은 본 논문에서의 기업 통합을 위한 인터넷 기반 프로세스 정보 인프라를 나타낸다. 예시된 바와 같이 제시된 Framework은 설계외에 타 부문으로 확장 가능하나 본 연구에서의 범위는 설계 부문을 위한 프로세스 관리의 적용에 한정한다.



<그림 1> 기업 정보 프로세스 구조

II. 제안 시스템 Approach

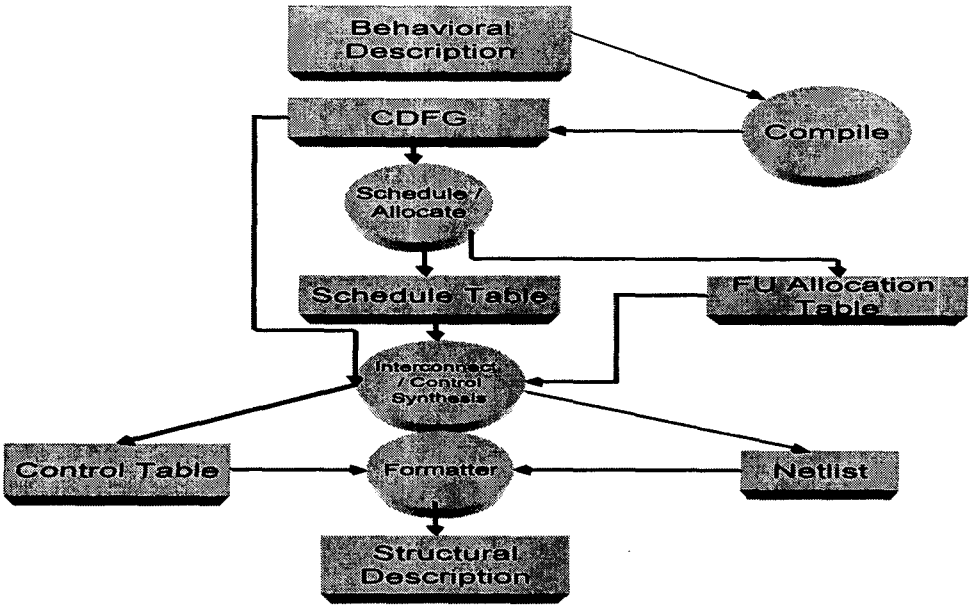
본 제안 시스템은 프로세스 흐름 그래프와 설계 프로세스 문법(Process Grammar)에 근거한 방법론을 사용한다. 프로세스 흐름 그래프(Process Flow Graph)는 설계 프로

세스의 정보의 흐름을 기술하며 설계 프로세스 문법은 상위 레벨의 프로세스 그래프를 세부적인 그래프로 전환하는 수단이 된다.

2.1 프로세스 흐름 그래프

프로세스 흐름 그래프는 양분되며 비 순환적인 그래프 (Bipartite Acyclic Directed Graph)로서 그래프는 두개의 다른 종류의 노드(Node)로 구성된다. 모든 에지(Edge)의 연결은 한 타입에서 다른 타입으로 연결되며 모든 통로(Path)에서 자신의 노드로 다시 유입되는 것은 없다.

본 논문에서는 노드를 태스크 노드(Task Node)와 상세 노드(Specification Node)로 분류하며 Task노드는 종료 노드(Terminal Node)와 비 종료 노드(Non-Terminal Node)로 나뉜다. 종료노드는 원격지 툴 실행(Tool Invocation) 노드로 애프리케이션 프로그램을 실행하게 되며 이중 원형으로 표현된다. 비 종료 노드는 소수개의 다른 툴 조합으로 분할 되어지는 추상적(Abstract) 태스크로서 단일 원으로 표현되어 진다. 각 세부 노드(Specification Node)에서 세부 사양 업무가 표시되고 각 사양 노드(Specification Nodes)는 한 개의 유입 에지 (Incoming Edge)를 갖는다. 유입 에지 (Incoming Edge)를 갖지 않는 사양 노드는 초기 입력 값을 의미한다. T(G), S(G)와 E(G)는 각각 태스크 노드, 사양 노드의 집합과 그래프 G의 에지를 표현한다.



<그림 2> Sample Process Flow Graph For High Level Synthesis

그림 2는 Rapid Prototyping 설계 프로세스를 표현한 것으로 Behavioral Description이 Structural Description으로 변형되는 프로세스 흐름 그래프의 예를 나타낸다. 다양한 세부사양 타입(Specification Type)은 클라스 Hierarchy를 구성한다. 노드는 Parent 노드는 몇 개의 각 Child 노드로 구성되고 Parent노드의 세부 사양을 나타낸다.

프로세스 흐름 그래프는 다양한 상세 레벨의 설계 프로세스를 표현 할 수 있는데 종료 노드가 어떠한 세부 툴이 사용되어져야 하는가를 나타낸다면 비 종료 노드는 상위레벨에서의 방법론을 표현하게 된다.

In(N) 이 노드 N의 입력노드 집합이면:

In(N) = { M | (M, N) E}.

Out(N) 이 노드 N의 입력노드 집합이면:

Out(N) = { M | (N, M) E}.

I(G)이 그래프 G의 입력 상세사양 집합이면:

I(G) = {N S (G) | In(N) = }과 같이 표현 되어 질 수 있다.

2.2 설계 프로세스 문법

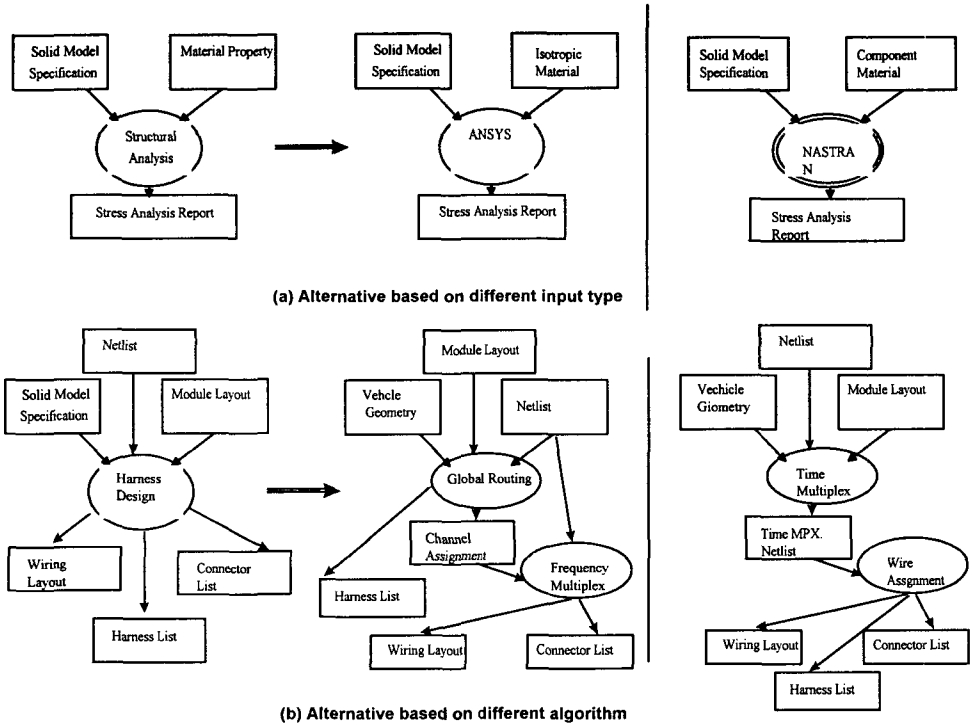
설계자는 사용 가능한 입력 상세사양, 요구되는 출력 상세사양과 수행되야할 추상 태스크 (Abstract Tasks)를 갖는 초기 그래프 (Initial Graph)의 전체적 목표를 설정하게 된다. 설계 Process Grammars에 의해 비 종료 태스크 노드는 좀더 세부적인 Abstract Task와 중간 세부사양으로 대치된다. 출력 세부사양 노드는 Child 세부사양 타입을 갖는 노드로 전환된다.

Graph Grammar의 프로덕션(Productions)에 의해 하나의 서브 그래프는 다른 서브그래프로 대치된다. 이러한 프로덕션은 다음과 같은 튜플(Tuple)로 표현 되어진다;

$$P = (G_{LHS}, G_{RHS}, {}_{In}, {}_{Out})$$

여기서 G_{LHS}과 G_{RHS}과는 각각 좌변과 우변의 프로세스 흐름 그래프를 나타낸다. 따라서 $T(G_{LHS})$은 변환되어져야 할 싱글(Single)의 추상 태스크를 나타내는 비 종료 노드 단수의 낸다. $_{In}$은 $I(G_{RHS})$ 에서 $I(G_{LHS})$의 매핑으로 입력 세부 사양간의 동치를 나타내며 $_{Out}$은 $S(G_{LHS})$-$I(G_{LHS})$ 에서 $S(G_{RHS})$으로의 매핑으로 입력 세부 사양간의 동치를 나타낸다.

그림 3은 Compile, Schedule 및Allocate 태스크들에 대한 프로덕션을 나타낸다. 매핑은 세부사양 노드들외에 숫자로도 표현된다. 3a는 다른 입력 사양 타입에 대하여 동일한 출력 타입을 산출하는 경우이며, 3b는 다른 알고리즘에 의해 다른 프로세스로 변형 가능한 프로덕션을 의미한다.



<그림 3> Two Types Of Production

A를 $T(G_{LHS})$의 비 종료 태스크 노드라 하고 A를 기존의 프로세스 흐름 그래프 G라 하면 프로덕션은 A와 다음과 같은 조건이 만족되면 매치가 된다.

A 가 A와 같은 태스크 라벨을 갖고,

In(A)로 부터In(A)로의 매핑 $_{In}$ 이 존재하며 모든 노드 N In(A)에 대해 $_{In}$(N)이 동일 타입 또는 서브 타입을 갖는다.

Out(A) 로 부터 Out(A)로의 매핑 $_{Out}$ 이 존재하며 모든 노드 N Out (A)에 대해 $_{In}$(N)이 동일 타입 또는 서브 타입을 갖는다.

매핑은 에지들이 어떻게 새 서브 그래프에서 재 구성 되야하는가를 결정하기 위해 사용되어 진다. 그래프 G에서 매치가능성이 있으면 프로덕션은 다음과 같이 적용된다.

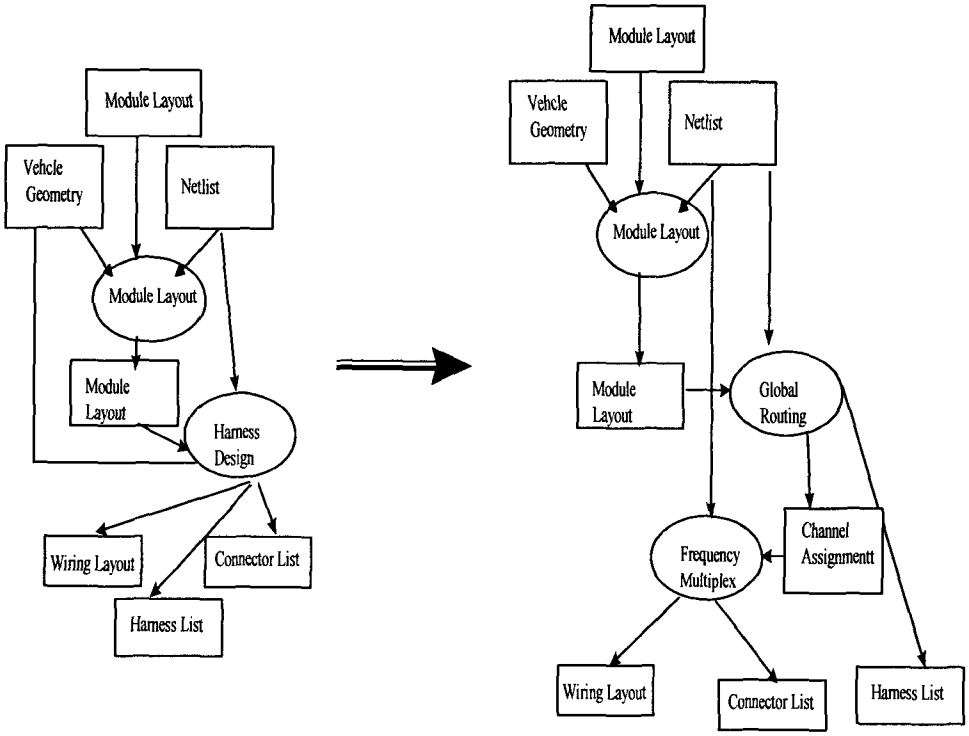
G에 G_{RHS} - $I(G_{RHS})$를 삽입한다. 대치될 태스크의 입력값은 변환되어 지지 않은 상태이다.

$I(G_{RHS})$에 모든 N과 G_{RHS} 의 Edge(N, M)에 대해 ($_{In}$ ($_{In}$(N)), M)를 그래프에G 부가한다.

Out(A)에 모든 N과 G의 Edge(N, M)에 대해 Edge (N, M)을 Edge ($_{Out}$ ($_{Out}$ (N)), M)로 대치한다.

G로부터 A 와 Out(A)를 제거한다.

그림 4는 3b에서의 프로덕션을 이용해 전환된 예를 나타낸다.



<그림 4> A Sample Graph Derivation

III. 실행 운영 환경

조직이 분산되어 있는 가상기업에서 실행환경은 프랫폼에 독립적 (Platform Independent) 이어야 한다. 설계 프로세스 관리를 위한 Framework 은 설계 프로세스의 정형화된 표현을 중심으로 Process Grammar를 통해 작업을 가능한 계층구조로 Decompose하고 이를 코드화 한다. 설계자는 설계 프로세스 Grammar로 부터 Process Flow Graph를 만들 수 있고 계층적인 설계 방법론을 자연히 얻어낼 수 있기 때문에 설계 공간(Design Space)을 체계적으로 탐색할 수 있도록 한다.

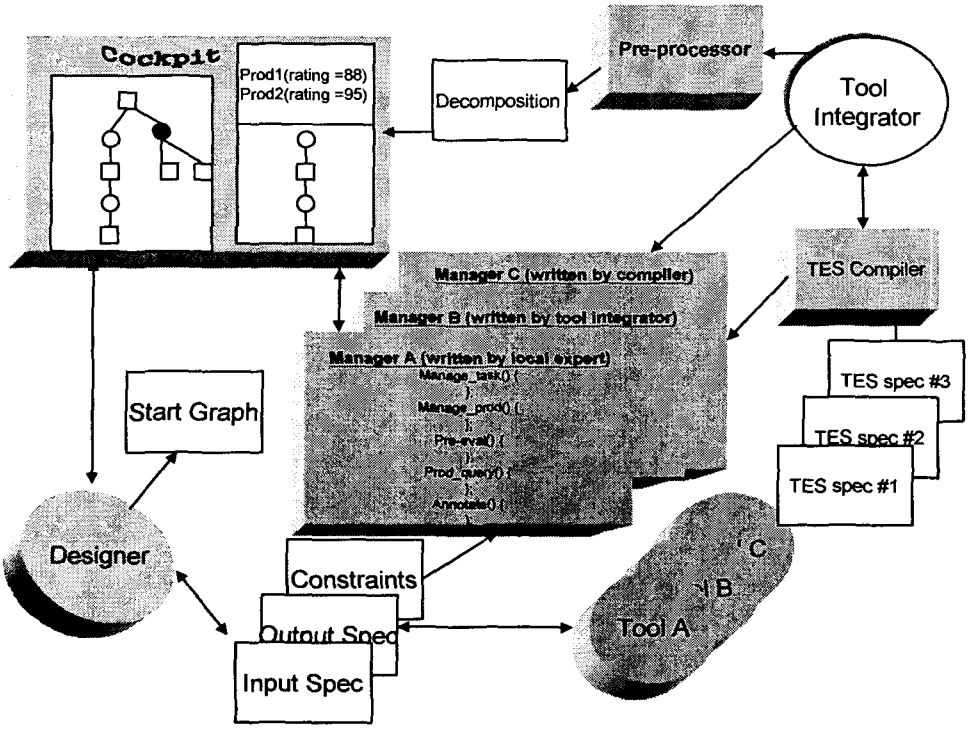
설계자와 시스템과의 관계는 Cockpit이라고 하는 프로그램으로 Interaction이 일어나도록 한다. 이 Cockpit은 현재 설계 상태를 계속적으로 추적하고 가능한 Action을 사용자에게 알려주며 설계자가 적절한 Action을 선택할 수 있도록 Manager Program과 서로 대화한다.

Manager Program들은 설계 Decomposition에 대한 평가를 하고 Tool을 Load하며 결과를 Check한다. Manager Program들은 어떤 특정 정보를 반영하기 위해 Tool Integrator라고

하는 것에 의해 유지된다. Tool Integrator는 Manager Program을 새롭게 만들어질 때 시스템과 연계를 위한 소스 코드를 자동적으로 만들기 위해 Tool Vendor에 의해 제공되는 정보를 이용한다. 또한 Tool Integrator에서 Cockpit에 입력 파일을 유지시킬 수 있도록 하기 위해 Pre-Process가 제공된다. 그림 5는 전술된 기능이 포함된 본 제안 시스템의 흐름도를 나타낸다.

분산된 객체들간에 커뮤니케이션은 객방형 산업 표준 (Industrial Open Standard)을 준수해야 한다..

그림 5는 전술된 문법을 적용한 본 제안 시스템의 윤곽을 나타낸다. 사용 툴의 선정과 실행은 루틴 결정을 하는 Manager Program과 상위레벨의 아이디어가 요구되는 설계자로 분류되어진다. Cockpit Program은 Manager Program과 설계자간에 상호작용을 관리 조정한다. Tool Sets과 방법론의 선택은 사용되어질 지역과 시간에 따라 달라 질 수 있다. 따라서 본 연구에서는 각 지역 사이트에서 Tool-Dependent Code를 작성하고 보수하는 System Integrator 역할을 하는 담당자가 있음을 전제로 하며 이를 위해 Tool-Independent Code 및 Templates를 지원한다.



<그림 5> System Overview

3.1 조정실

Cockpit은 설계 프로세스의 현 상태를 추적 관리한다. 따라서 Manager Program과는 다르게 특정 태스크에 대한 지식은 없다. 설계 프로세스의 모든 정보는 비 종료 노드에 대한 가능한 태스크와 Decomposition의 집합을 나타내는 Process Grammar에 따르는 입력 파일에 내포되어 있다. Cockpit은 Text Editor에서 산출된 입력 파일로부터 Initial Process Flow Graph를 읽고 언제 비종료 노드에 프로덕션이 적용 될 수 있는가를 계속적으로 나타내면서 Production Manager에게 Constraints에 근거한 Rating을 요구하고 적절한 Production이 적용되도록 한다.

Process Flow Graph와 가능한 프로덕션에 Rating이 설계자에게 보여지면 설계자는 Cockpit이 프로덕션과 Task를 특정 노드에 적용하도록 한다. 그러면 Cockpit은 Task Manager에게 메세지를 보낸다. 종료태스크 노드의 경우는 해당 툴에 실행 시키며 비 종료태스크 노드의 경우에는 Task Manager에 의해 하나 이상의 프로덕션을 적용하게 된다. 즉 Cockpit은 프로덕션을 적용하고 프로덕션 Manager가 이를 실행하도록 한다.

다음은 Cockpit의 알고리즘을 나타낸다.

```
Initialization()
{
   Start Graph Is Selected;
   Create Initial Daemon Process And Place Tokens (Send Message);
}
Wait For Message;
   IF The Message Is From Execution Process THEN
   {
      IF The Message Is TO_ELABORATE THEN
      {
                  Invoke Pre-Evaluation Function;
         Select Production Based On Pre-Evaluation;
         Display Expanded Process Flow;
         Send ELABORATE_THIS Or FAILED Message To Execution Process;
      }
      IF The Massage Is POST_EVAL THEN
      {
         Post-Evaluation;
         Send Result POST_EVAL_OK Or _FAIL To Execution Process;
      }
      IF The Message Is FAIL_ELABORATE THEN
         Delete Useless Tokens;
```

```
        IF The Message Type Is FAILED THEN
           Kill The Child Process;
        }
ELSE /* Message From Daemon Process */
           {
                   IF The Message Is FAILED THEN
                              Kill The Child Process;
                              ELSE
                              {
                              Create A Daemon From The Subsequent Task;
        Put The Output Token In The Newly Created In The Newly Created DaemonS Input
        Place;
                              }
}
END IF;
```

3.2 관리 프로그램

Manager Programs 은 해당 태스크에 대한 Knowledge Source를 갖으며 새로운 툴이 첨가되고 실행 중 얻은 경험을 일반화하여 지식을 Upgrade한다. 각각의 Manager Program은 다음과 같은 5개의 기능을 갖는다.

사전 평가(Pre-Evaluation)
툴 실행(Tool Execution)
추상 태스크의 실행(Abstract Task Execution)
프로덕션 실행(Production Execution)
질의 처리(Query Handling)

사전 평가(Pre-Evaluation): Production Manager는 설계자에게 각 Production에 대해 Rating을 주고 Task Manager가 최적의 프로덕션을 선정하도록 한다. Rating은 각 프로덕션에 대해 1과 100 사이의 성공 정도를 수치화 한것으로 System Integrator에 의해 사전 정의된다. 설계자는 프로덕션에 대한 변수를 정의하여 ASCII 포맷 형태로 표기한다. 다음은 Production Rating의 한 예를 보여준다.

```
Arch_Syn. PRE ./Preeval1
Arch_Syn. POST ./Hello
Arch_Syn. 0 Time 9 Pref 3 History 4
```

Arch_Syn. 1 Time 11 Pref 3 History 5
Arch_Syn. 2 Time 4 Pref 6 History 2

Production Arch_Syn 이 적용되면 사용자는 현재의 사용 디렉토리에 있는 Preeval1 이라는 Pre-Evaluation Function을 사용한다. 마지막 세개 라인은 실제 변수와 사용자에 의해 지정된 값들이 된다. 설계자는 이 값을 이용해 Pre-Evaluation Function 을 작성한다. 간단한 예는 다음과 같다.

```
#Include <Stdlib.H>
#Include <Stdio.H>
Main (Int Argc, Char **Argv)
{
Int T, P, H;
Int Score;
If (Argc < 7)
   Exit (-1)
T=Atoi (Argv[2]);
P=Atoi (Argv[4]);
H=Atoi (Argv[6]);
Score = T*0.2 + P*0.1 + H*0.7;
Exit(Score);
}
```

여기서 T, P 및H는 각각 해당 Production 종결 시점, 변환되어 지는 Input File Type에 대한 Penalty 값을 갖는 변수, 해당 Production 이 사용 가능 했던 가간을 의미한다.

정적 (Static) Rating은 해당 Task Node에 비성공적인 전례가 있으면 조정되어 질 수 있다. Rating은 질의 기능이나 입력 파일의 분석에서 얻어진 파라메타일 수 도 있다. Manager Program은 계속적으로 성공 조건을 나타내는 프로세스 행렬을 수집 분석할 수 있다.

Tool Execution: 종료 Tasks는 해당 Task Managers에 의해 소프트웨어 툴 실행을 하고 그 성공 여부를 판단한다. 대부분의 경우 정보는 사전정의 되고 표준 Template으로 입력된다. 다른 경우는, Manager는 Task-Specific Knowledge을 검사하여 Tool Parameters를 결정하거나 Task-Specific Constraints을 확인하여 성공여부를 결정한다.

Abstract Task Execution: 비종료 Tasks에 대해 Task Managers는 Abstract Task 를 실행할 프로덕션을 선택한다. Cockpit은 Task Manager에 사용가능한 프로덕션과 해당 Rating을

알려준다. Task Manager은 Cockpit이 한 개 이상의 프로덕션을 적용 및 실행하도록 하거나 실패 여부를 알려준다. 만일 해당 Production 이 성공하면 Task Manager는 Constraints을 확인한 후 만족되어지면 성공 메시지를 나타낸다.

Production Execution: Production Managers는 프로덕션의 우편(Right-Hand Side)에 해당하는 태스크들을 실행한다. 만일 이들 중 하나의 태스크가 구속조건에 위반되면 Backtracking이 된다. Production Manager 는 특정 업부에 관련된 지식을 가지고 있으며 Production Manager가 실패(Failure)를 처리 할 수 없을 경우에는 이를 Cockpit에 알리고 보다 상위 레벨의 Task Manager가 이를 처리한다.

Query Handling: Production Manager와 Task Manager들은 질의 메커니즘에 관여되어 이들은 상위 Parent Manager에게 또는 하위 Child Manager에게 질의를 하게 된다.

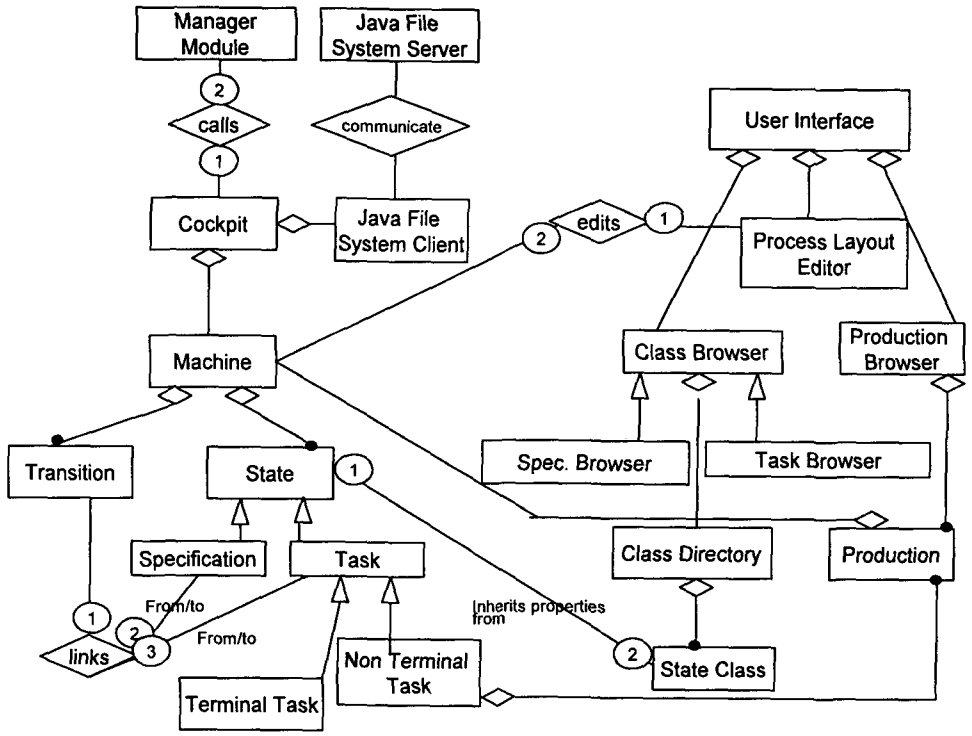
Ⅳ. 구 현

제안된 프로세스 관리 시스템은 객체지향 방법론을 이용하였다. 설계 프로세스 관리 시스템은 Internet을 기반으로 하는 Network환경에서 개발되었다. Process Flow, Process Grammar 및 Process관련 지식을 모델링하기 위해 OMT (Object Method Technology) 객체지향기법이 이용되었으며 개발 언어는 Web Program을 위해 Java, Perl 및 Tcl/Tk가 이용되고 기타 모듈은 C++로 개발되었다.
본 절에서는 설계 프로세스 관리 시스템 구현의 주요 기반이 되는 클라스 정의에 대해 알아본다.

그림 6은 OMT로 표현된 설계 프로세스 관리시스템의 상위 레벨 객체 모델로서 각 객체간에 관계성을 나타낸다[Rumbaugh Et Al., 91].

Cockpit: Cockpit은 Class *COCKPIT* 으로 시작되고 Class *PRMAIN* 은 설계 프로세스 관리 시스템에 대한 Control Point이다. 시스템이 시작될 때 Cockpit은 Root Process Flow가 되는 Class *MACHINE* 인스탄스를 산출한다. Cockpit의 가장 중요한 기능은 프로세스 시뮤레이션 제어로서 Cockpit은 *STATE* Object들 Rollback 및 Pre- And Post-Evaluation을 담당하는 다양한 Manager Modules과의 Communication을 관리 통제한다.

마지막으로 Java File System 서버와의 연계를 제어하여 특정 프로세스가 저장 및 로딩 (Loading) 되도록 한다.



<그림 6> High-Level Object Model Of Process Management System

Manager Modules: 이는 COCKPIT Class가 연계를 갖는 외부 프로그램이다. 이는 특정 NON-TERMINAL-TASK, 에 대해 모든 적용 가능한 PRODUCTIONS을 평가 하고 각 프로덕션에 대한 Rating을 돌려준다(Pre-Evaluation 단계). 또한 Managers는 PRODUCTION의 출력 값을 평가하게 된다 (Post-Evaluation 단계).

Java File System: Java 파일 시스템 (JFS)는 Java Applet이 생성된 서버에 파일을 저장하게 한다. 이는 클라이언트 / 서버 구조를 갖는다. COCKPIT Class는 실행되는 JFS Server와 Communication되는 JFS Client를 유지 관리한다.
Machine: MACHINE Class는 설계 프로세스 관리 시스템의 가장 중심이 되는 객체중 하나로 TRANSITIONS 과 STATES 의 벡터를 유지관리 함으로서 프로세스 흐름(Process Flow)을 표현 한다.

Transition: TRANSITION Class는 STATE Class와 논리적 연결(Logical Link)을 갖으며 TRANSITION Class의 Instantiation은 SPECIFICATION에서 STATE로 혹은 STATE에서 SPECIFICATION로 연계된다.

State: STATE Class는 SPECIFICATION And TASK 의 상위 Class로 MACHINE 이 Finite Automaton로 표현 되는 반면 STATE 는 Finite Automaton에서 Generic Node를 나타낸다.

Specification- SPECIFICATION 은 TASK 의 입 출력 역할을 하는 STATE의 한 타입이다.

TASK는 TERMINAL-TASK 과 NON-TERMINAL-TASK 두 서브 타입의 상위 Class 이다.

NON-TERMINAL-TASK Class 는 추상 태스크(Abstract Task)를 나타내어 At 적어도 한 개 이상의 PRODUCTION 인스탄스를 갖는다. TERMINAL-TASK 는 실제 툴과 연계된다. TERMINAL-TASK 가 Process Simulation이 될 경우 원격지의 툴은 실행(Invocation)이 된다.

User Interface:

Process Layout Editor- 몇 개의 AWT 그래프 클라스로 구성되어 있고 MACHINE을 구성하는 STATES 와 TRANSITIONS로 조정 관리하기 위한 기능을 갖는다.

Class Browser- 설계 프로세스 관리 시스템에서는 Task Browser와 Specification Browser등의 두개 Class Browsers 가 제공 되는데 이들은 AWT 구성요소와 동일하며 STATE-CLASS 구조를 유지하기 위한 CLASS-DIRECTORY 의 인스탄스를 사용한다.

Production Browser- Production Browser는 모든PRODUCTION과 각 PRODUCTION이 포함되는 NON-TERMINAL-TASK 의 벡터를 유지 관리 하며 사용자에게 Process Layout Editor를 이용하여 각 PRODUCTION 을 조정하기 위한 AWT 기능들을 제공한다.

Production: 모든 NON-TERMINAL-TASK 은 적어도 하나의 PRODUCTION 클라스에 인스탄스를 갖는다. 각 PRODUCTION 은 특정한 NON-TERMINAL-TASK에 대한 하나의 가능한 대안 프로세스 프로우를 나타낸다. 프로세스 프로우의 유지는 MACHINE 클라스의 인스탄스를 통해 이뤄진다.

Class Directory: Class Browsers는 STATE-CLASS 객체의 벡터를 유지하기 위한 STATE-CLASS-DIRECTORY Class 의 인스탄스를 사용한다.

V. 결론

본 논문에서는 기업 통합 구현을 위한 가장 중요한 요소 중의 하나인 설계 프로세스 관리를 위해 Internet 환경 하에서 다양한 설계 요구조건을 만족하여 최적 해를 얻을 수 있는 Framework을 소개하였다. 프로세스 관리 시스템에서는 설계 프로세스를

표현하기 위한 Process Grammar를 개발하고 관련 데이터들 간의 일관성을 유지하면서 설계 프로세스에서 발생되는 Activity들의 관리 통제 기능을 개발하였다.

참 고 문 헌

[1] Michael L. Bushnell and S.W. Director. VLSI CAD Tool Integration Using the Ulysses Environment. 23re ACM/IEEE Design Automation Conference, pp 55-61, 1996.

[2] Andre Casotto, A. Richard Newton, and Alberto Sangiovanni-Vincetelli. Design Management based on Design Traces. 33th ACM/IEEE Design Automation Conference, pp 136-141, 1997.

[3] Tzi-cker F. Chiueh and Randy H. Katz. A History for Managing the VSLI Design Process. International Conference on Computer Aided Design, pp 358-161, 1990.

[4] Reid Baldwin, Sea H. Choi, and Moon J. Chung. VHDL Synthesis Framework. Spring VIUF 94, May 1994.

[5] Reid A. Baldwin and Moon Jung Chung. A Formal Approach to Managing Design Processes. IEEE Computer, pp 54-63, February 1995.

[6] K.O. ten Bosch, P. Bingley, and P. van der Wolf. Design Flow Management in the NELSIS CAD Framework. Proceedings of the 28th Design Automation Conference, pp 711-716, 1991.

[7] James Daniell and Steven W. Director. An Object Oriented Approach to CAD Tool Control. IEEE Transactions on Computer-Aided Design, pp 698-713, 1991.

[8] Alberto Di Janni. A Monitor for Complex CAD Systems. Proceedings of the 23rd Design Automation Conference, pp 145-151, 1996.

[9] Douglas G. Fairbairn. 1994 Keynote Address. Proceedings of the 31the Design Automation Conference, pp xvi-xvii, 1994.

[10] Meichun Hsu and Charley Kleissner. ObjectFlow: Towards a Process Management Infrastructure. Distributed and Parallel Databases, 4:169-194, 1996.

[11] Margarida F. Jacome and Stephen W. Director. Design Process Management for CAD Frameworks. Proceedings of the 29th Design Automation Conference, pp 500-505, 1992.

[12] Randy H. Katz, Rajiv Bhateja, Ellis E-Li Chang, David Gedye, and Vony Trijanto. Design Version Management. IEEE Design and Test, 4(1): 12-22, February 1987.

[13] David Knapp and Alice Parker. The ADAM Design Planning Engine. In AI in Design, Vol. II, pp 263-285. Academic Press, 1992.

[14] James Rumbaugh et al. Object-Oriented Modeling and Design. pp 57-79. Prentice Hall, 1991.

[15] Daniel D. Corkill, Kevin Q. Gallagher, and Philip M. Johnson, Achieving Flexiblility, Efficiency, and Generality in Blackboard Architectures, Proceedings of the National Conference on Artificial Intelligence, 1997, pp18-23

[16] Hwa Gyoo Park, Feature Recognition System Based on Neural Network Technique, Proceedings of 4World Congress on Expert Systems, 1998.

[17] Hwa Gyoo Park, Expert Classification System for Concurrent Engineering, Proceedings of Concurrent Engineering 98 Japan, 1998.

<Abstract>

Internet Based Managing Design and Production Processes in a Distributed Global Environment

Hwa Gyoo Park

This paper is to develop an information infrastructure to support managing process in design, planning, production, and quality control. Multi-media data set of design, product, and management information flow between organizational units of a virtual enterprise. The process is the logical organization of people, technology and practices incorporated into work activities to make an end product. The core of the infrastructure is the enterprise framework which coordinates activities and controls the process. The proposed framework manages collaborative activities across space and time, and between users and computers who share information in virtual community. It utilizes knowledge distributed through virtual community and fosters cooperation between organizations. The framework provides the following facilities; coordinating activities, sharing data and processes, visualizing multi-media data, customizing and updating processes, reusing data and processes. This paper covers design and manufacturing activities but our focus is initially targeted at design area.